(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 March 2006 (02.03.2006)

PCT

(10) International Publication Number
WO 2006/022745 A2

(51) International Patent Classification: *G06F 9/445* (2006.01)

(21) International Application Number: PCT/US2004/028195

(22) International Filing Date: 30 August 2004 (30.08.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/926,635 24 August 2004 (24.08.2004) US

(71) Applicant *(for all designated States except US)*: STREAM THEORY, INC. [US/US]; 3350 Scott Boulevard, Building 24, Santa Clara, CA 95054 (US).

(72) Inventors: DE VRIES, Jeff; 902 West Olive Avenue, Sunnyvale, CA 94086 (US). HUBBELL, Ann; 26530 Conejo Court, Los Altos Hills, CA 94022 (US). ZAVERTNIK, Greg; 2983 Calle de las Estrella, San Jose, CA 95148 (US).

(74) Agent: COLEMAN, Brian, R.; Perkins Coie LLP, 101 Jefferson Drive, Menlo Park, CA 94025 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: INTERCEPTION-BASED RESOURCE DETECTION SYSTEM

(57) Abstract: A technique for resource detection involves running an application installer associated with an application and intercepting calls made by the application installer. A system constructed according to the technique can infer what resources are required by the application from the intercepted calls. The system may generate streaming files to facilitate streaming the application associated with the installer to a remote location.

WO 2006/022745 A2

25

INTERCEPTION-BASED RESOURCE DETECTION SYSTEM

BACKGROUND

Virtual installation is used to facilitate streaming of software. With virtual installation, software can be virtually installed without actually downloading the software. One technique for virtually installing software involves streaming. Typically, software streaming entails anticipating what files and resources are to be consumed by a program during the streaming. A technique for ensuring that the software has the required resources involves taking a snapshot of a drive before installing a program, installing the program, taking a snapshot after installing the program, and comparing the snapshot before and after the installation to determine what changes have been made to the drive. In this way, the resources to be used by the program can be determined.

However, this technique requires taking snapshots of a drive, which can be time-consuming and may consume relatively large amounts of memory or storage resources.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 depicts a networked system for use in an embodiment;

FIG. 2 depicts a computer system for use in the system of FIG. 1;

FIG. 3 depicts a portion of the computer system of FIG. 2 and components of the system of FIG. 1;

FIGS. 4A to 4C illustrate the capture of calls according to an embodiment;

FIGS. 5A and 5B depict data that is recorded in an embodiment;

FIG. 6 depicts a flowchart of an exemplary method for interception-based resource detection;

FIGS. 7A-7K depict screenshots intended to illustrate an exemplary GUI according to an embodiment;

FIG. 8 depicts a flowchart of an exemplary method for jumpstart;

FIG. 9 depicts a flowchart of an exemplary method according to an embodiment.

Internet allows users of the client computer systems to exchange information, receive and send e-mails, and view documents, such as documents which have been prepared in the HTML format. These documents are often provided by web servers, such as web server 104, which are referred to as being "on" the Internet. Often these web servers are provided by the ISPs, such as ISP 110, although a computer system can be set up and connected to the Internet without that system also being an ISP.

Client computer systems 112, 118, 122, and 126 can each, with the appropriate web browsing software, view HTML pages provided by the web server 104. The ISP 110 provides Internet connectivity to the client computer system 112 through the modem interface 114, which can be considered part of the client computer system 112. The client computer system can be a personal computer system, a network computer, a web TV system, or other computer system. While Fig. 1 shows the modem interface 114 generically as a "modem," the interface can be an analog modem, isdn modem, cable modem, satellite transmission interface (e.g. "direct PC"), or other interface for coupling a computer system to other computer systems.

Similar to the ISP 114, the ISP 116 provides Internet connectivity for client systems 118, 122, and 126, although as shown in Fig. 1, the connections are not the same for these three computer systems. Client computer system 118 is coupled through a modem interface 120 while client computer systems 122 and 126 are part of a LAN 130.

Client computer systems 122 and 126 are coupled to the LAN 130 through network interfaces 124 and 128, which can be ethernet network or other network interfaces. The LAN 130 is also coupled to a gateway computer system 132 which can provide firewall and other Internet-related services for the local area network. This gateway computer system 132 is coupled to the ISP 116 to provide Internet connectivity to the client computer systems 122 and 126. The gateway computer system 132 can be a conventional server computer system.

Alternatively, a server computer system 134 can be directly coupled to the LAN 130 through a network interface 136 to provide files 138 and other services to the clients 122 and 126, without the need to connect to the Internet through the gateway system 132.

FIG. 2 depicts a computer system 140 for use in the system 100 (FIG. 1). The computer system 140 may be a conventional computer system that can be used as a client computer system or a server computer system or as a web server system. Such a computer system can be used to

often have multiple buses, one of which can be an I/O bus for the peripherals and one that directly connects the processor 148 and the memory 152 (often referred to as a memory bus). The buses are connected together through bridge components that perform any necessary translation due to differing bus protocols.

Network computers are another type of computer system that can be used with the present invention. Network computers do not usually include a hard disk or other mass storage, and the executable programs are loaded from a network connection into the memory 152 for execution by the processor 148. A Web TV system, which is known in the art, is also considered to be a computer system according to the present invention, but it may lack some of the features shown in FIG. 2, such as certain input or output devices. A typical computer system will usually include at least a processor, memory, and a bus coupling the memory to the processor.

In addition, the computer system 140 is controlled by operating system software which includes a file management system, such as a disk operating system, which is part of the operating system software. One example of an operating system software with its associated file management system software is the family of operating systems known as Windows® from Microsoft Corporation of Redmond, Washington, and their associated file management systems. Another example of operating system software with its associated file management system software is the Linux operating system and its associated file management system. The file management system is typically stored in the non-volatile storage 156 and causes the processor 148 to execute the various acts required by the operating system to input and output data and to store data in memory, including storing files on the non-volatile storage 156.

Some portions of the detailed description are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of operations leading to a desired result. The operations are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

include the network 162. In another alternative, the computer system 140 may by physically or wirelessly coupled to the testing computer 164.

It had been discovered through trial-and-error that a pristine operating environment is important for testing of certain applications, such as streaming applications. To this end, in an embodiment, ghosting is used to ensure a pristine testing environment. Ghosting is a technique that includes preparing a clean environment, creating a copy of a partition associated with the clean environment (the ghosted partition), and overwriting a partition with the ghosted partition to restore the clean environment. Thus, ghosting could be thought of as a "partition, save, restore" procedure. One utility that facilitates ghosting is Norton Ghost™ 7.5. In an embodiment, simply removing applications or performing a conventional backup is insufficient to remove certain files or registry values. For example, performing a conventional backup may remove an application, but leave a registry value related to the application. The registry value may be used again if the application is re-installed. However, in the context of a streaming application, a testing environment must be started anew each time as if the streaming application had never been on the testing machine. Accordingly, in an embodiment, the testing computer 164 may have a ghosted configuration. Alternatively, the testing computer 164 may have a clean operating environment that is accomplished by deleting files and registry value artifacts associated with an application that remain after the application is removed.

The testing computer 164 may be coupled to and accessible through the network 162. The testing computer 164 could be part of a computer system, such as the computer system 140 (FIG. 2). The testing computer 164 may include a processor, a memory, and a bus that couples the processor to the memory. For the purposes of testing, a configuration should include as few irrelevant or distracting applications as possible. Moreover, a testing computer 164 is often desirable in addition to the computer system 140 because artifacts may remain on the computer system 140 after processing an application. For example, if the computer system 140 is used to prepare an application for streaming, the preparation may spawn a registry value that is not incorporated into the streaming application. Since the registry value exists on the computer system 140, the streaming application, if tested on the same computer, may make use of it even though the registry value was not properly captured and incorporated into the streaming application. Then, when the streaming application is run on a different computer, the streaming application may not have the appropriate registry values available to it. For this reason, in an

The installation of the program may be executed as a sub-process of the installer 174. Another example of a sub-process may include an installer 174 that gathers information from a user in a GUI then launches an installer (sub-process) that makes use of the information. Another example is the Microsoft™ installer, which runs under a service model; the subsystem copies files and creates registry entries. These are only examples. There are a number of different reasons and programming styles that may cause the installer 174 to be divided into multiple sub-processes. The sub-processes may or may not be installers themselves. For example, a sub-process could be invoked by the installer 174 or the sub-process could be invoked directly. A technique is provided herein that enables tracking of activities of the installer 174 and sub-processes of the installer 174. An example of a multi-process installer is described later with reference to FIG. 4B.

The interception-based resource detection program 172 can be logically divided into functional modules. For example, the interception-based resource detection program 172 may include an interception module 178, an application installer execution module 180, a recording module 182, a verification module 184, a parsing module 186, and a streaming application creation module 188.

In an embodiment, the interception module 178 is started prior to executing the installer 174. In an alternative, the interception module 178 could be started at the same time as, or after, the installer 174. The interception module 178 operates in runtime.

The application installer execution module 180 may provide a portion of an environment in which to execute the installer 174. In an embodiment, the interception-based resource detection program 172 provides at least part of the environment. In another embodiment, some or all of the application installer execution module 180 is provided by an operating system or a program for executing application installers, such as the installer 174. In an embodiment, the application installer execution module 180 includes a GUI in which a user may indicate that the installer 174, for example, is to be executed. The application installer execution module 180 then launches the installer 174.

While the installer 174 is running, interceptors associated with the interception module 178 capture calls made by the installer. The calls may include file system, registry, and other calls. It should be noted that the interception of calls may be accomplished by, for example,

FIG. 4C depicts the installer 192 (FIG. 4A) and a system service 206 that is invoked by the installer 192 with an API call. The system service 206 may or may not have a hierarchical relationship with the installer 192. Nevertheless, it may be desirable to capture (196) the calls associated with the system service 206. In an embodiment, to ensure that the calls associated with the system service 206 are captured, a table of names is maintained by, or for use by, the tool. The table of names includes a listing of possible system services (e.g., MSI.exe). When a system service that is listed in the table is executed in the context of an installation (e.g., while the installer 192 is running), that system service is assumed to be relevant to the installation. Accordingly, activity by the system service is captured. In an alternative, the interceptors 194 (FIG. 4A) could capture the API call itself and determine that the system service is associated with the installer 192 in the context of the installation.

Referring once again to FIG. 3, the recording module 182 may record installation-related data and infer which resources are to be associated with the installer 174 based upon, for example, the intercepted calls. The recorded data may include directories, files, registry keys, registry values and other data. The recording module 182 may record data as it is captured by the interception module 178 or after the installer 174 has finished. In an alternative, some recording functionality of the recording module 182 is included in the interception module 178. For example, the interception module 178 may both intercept and record calls. In this context, the interception and recording of data may be referred to as capturing data.

FIGS. 5A and 5B depict data that is recorded by the recording module 182 in an embodiment. As depicted in FIG. 5A, data may include a structure 208 that includes keys and values. The structure 208 is intended to represent registry keys and registry values. The structure 208 may be stored in a tracefile that includes a path to registry values that are created, modified, or deleted while the installer is executed. As depicted in FIG. 5B, data may include a structure 210 that includes directories and files. The structure 210 is intended to represent a file directory that may be stored in a tracefile that includes a path to files that are created, modified, or deleted while the installer is executed. In an embodiment, the tracefile associated with the file directory does not include the actual files (e.g., the tracefile includes only the path). Alternatively, the tracefile could include the actual files.

Referring once again to FIG. 3, in an embodiment, the recording module 182 maintains one or more working files or databases while recording data. These files may be tracefiles that

and a stream file (e.g., a .stc file). The streaming application may also include a text file (e.g., a .txt file). The streaming application creation module 188 may create and encode the token file. The token file includes paths to various resources. The token file may be thought of as containing the information necessary to trick a computer that is receiving a streaming application into believing it has all of the resources it needs to run the application. The streaming application creation module 188 may create and process (e.g., compress or encrypt files) the stream file. The stream file includes a series of blocks. The blocks are processed from a list of files that are segmented into blocks. When streaming an application to a computer, the computer may need some of the blocks at any given time. If the computer needs an additional block, the computer may request the additional block from a server and the server will provide the additional block from the stream file. A .stw file may also be created that includes a copy of the record. The .stw file can be merged with subsequent installations of updates or other applications, as described with reference to FIG. 6.

FIG. 6 depicts a flowchart of an exemplary method for interception-based resource detection. The flowchart starts at block 212 with running a detection program. The detection program may be an interception-based detection program, such as the interception-based detection program 172 (FIG. 3). The detection program may be run in the foreground or the background. Alternatively, the detection program could be run after first determining that an installer is going to be executed. The detection program may be locally available, or remotely available for download or streaming.

FIG. 7A depicts an exemplary GUI for use with the detection program described with reference to FIG. 6. In the example of FIG. 7A, an untitled application window is displayed, but no data has been entered (e.g., because an application installer has not yet been selected). In the example of FIG. 7A, the detection program is associated with StreamWeaver™ 3.0.

Referring once again to FIG. 6, the flowchart continues at block 214 with starting a monitor. The monitor may be, for example, an interception module, as described with reference to FIG. 3.

The flowchart continues at block 216 with executing an application installer. The application installer is associated with one or more applications. The application installer may be executed from within a window or shell associated with the detection program. For example,

Change to the files may or may not be reflected in a system directory. Some or all of the values may be incorporated into a token file as described later.

If, for example, the registry option (FIG. 7E) is selected, then a view such as depicted in FIG. 7G may be displayed. In FIG. 7G, the registry includes multiple values associated with the application. It should be noted that the value names and associated values may be changed and new values can be added. The data represented in the view of FIG. 7G represents what was initially installed by the installer. Change to the values may or may not be reflected in a system directory. Some or all of the values may be incorporated into a token file as described later.

If, for example, the project setting option (FIG. 7E) is selected, then a view such as depicted in FIG. 7H may be displayed. In FIG. 7H, various application and system information is displayed. A user may be required to enter some or all of the data, or some or all of the data may be entered automatically. In the example of FIG. 7H, the view includes an application information pane, an application launch pane, an operating system support pane, and a required system components pane.

In the application information pane, general information about a title may be specified. This information may be stored within the record (e.g., a .stw file) and may be used to track processed titles. Some of the information may also be used to generate stream and token files, as described later. The short name is used in naming the token and stream files and the short name determines the title name the end user sees on a software player, as described later. The information may also be used to create an application install file for a server, as described later. Most of this information may come from the publisher of the application, but Program Description (text) field may include comments about the application that may, for example, be entered by a processing organization. This information may or may not be included in the record for informational purposes only (e.g., the information would not be used to generate a stream file). This information may also be pulled into a database to manage the processed applications.

In the application launch pane (FIG. 7H), fields may include a command line that starts the application and a working directory from which the application locates associated files. Typically, the working directory is the same as the directory used in the command line, but this is not always the case.

naming convention: "app_name"_"uuid".stc, "app_name"_"uuid".tok, and "app_name"_"uuid".txt. The app_name may be the short name described above with reference to FIG. 7H. The uuid value is used to create uniquely identifiable files for each build. In an embodiment, the .stc file is put on a stream server and used to stream the associated application. In an embodiment, the .tok file is similar to the .tok file that is put on a token server. In an embodiment, the .txt file is used to install the application onto a server.

Referring once again to FIG. 6, if there are no additional updates (226-N) the flowchart continues at optional block 230 with copying the streaming application to a testing computer. Block 230 is optional because a testing computer need not be used. For example, the streaming application could be tested on the same computer that executes the installer or performs processing.

The flowchart ends at optional block 232 with testing the streaming application. Block 232 is optional because testing is optional (though recommended). When testing, a user may execute the .tok file to start a software player and the associated application. The application can be tested to verify that it works correctly. The application may then be jumpstarted.

Jumpstarting is a process of specifying which stream blocks of the application need to be in a streaming software player cache before the application starts to run. A variable amount of data may be cached for jumpstarting. In general, if a relatively large amount of data is pre-cached, first-run start times tend to be faster and later delays as blocks are loaded on demand tend to be reduced. Note that the second time the application is run, unless flushed from the cache, application start-up times may be faster given that the application is loaded from the disk cache.

FIG. 8 depicts a flowchart of a method for performing jumpstart. The flowchart starts at block 234 with generating jumpstart data. Jumpstart data may be generated by, for example, executing a streaming application for a period of time. In the context of games, jumpstart data could be generated by completing a level (e.g., “Level One”) of a game. The jumpstart data identifies the blocks that were used when generating the jumpstart data. For example, if when generating the jumpstart data, a computer accesses blocks 1, 2, and 3 of the streaming application (e.g., of a .stc file), then the jumpstart data will include an identifier or pointer to the blocks 1, 2, and 3. Later, when the streaming application is jumpstarted, it is known that blocks

thereof are within the inventive scope of the present invention. It is therefore intended that the following appended claims include all such modifications, permutations and equivalents as fall within the true spirit and scope of the present invention; the invention is limited only by the claims.

a means for executing an application installer, wherein the application installer is associated with one or more applications;

a means for intercepting calls made by the application installer;

a means for determining from the intercepted calls resources and settings to be associated with the applications; and

a means for creating a stream file that, when run, is configured to stream the applications.

6. A system comprising:

an application installer execution module configured to at least partially provide an environment in which an application installer is executed, wherein the application installer is associated with one or more applications;

an interception module configured to intercept calls made by the application installer while executed in the environment;

a recording module configured to record installation-related data and infer resources to be associated with the applications from the intercepted calls; and

a stream file creation module configured to create a stream file that, when run, is configured to stream the applications.

7. The system of claim 6, further comprising:

a means for associating an update with the applications;

a means for executing an update installer associated with the update;

a means for intercepting calls made by the update installer;

a means for determining from the intercepted calls resources and settings to be associated with the update; and

a means for updating the streaming application according to settings and resource requirements of the update.

8. The system of claim 6, further comprising:

a means for creating an initial streaming application;

a means for capturing jumpstart data; and

a means for merging the jumpstart data into the initial streaming application.

9. The system of claim 6, further comprising:

jumpstart data, wherein the streaming application creation module merges the jumpstart data into the initial streaming application.

16. The system of claim 13, wherein when jumpstart data for logical subdivisions of a streaming application has been captured, the streaming application creation module facilitates merging the jumpstart data for each of the logical subdivisions and rebuilding the streaming application using the merged jumpstart data.

17. The system of claim 13, wherein the calls made by the application installer are associated with modification of a registry.

18. The system of claim 13, wherein the calls made by the application installer are associated with modification of a file.

19. The system of claim 13, further comprising a streaming application creation module that determines from the intercepted calls resources to be associated with the applications.

20. The system of claim 13, wherein the streaming application includes a token file and a stream file.

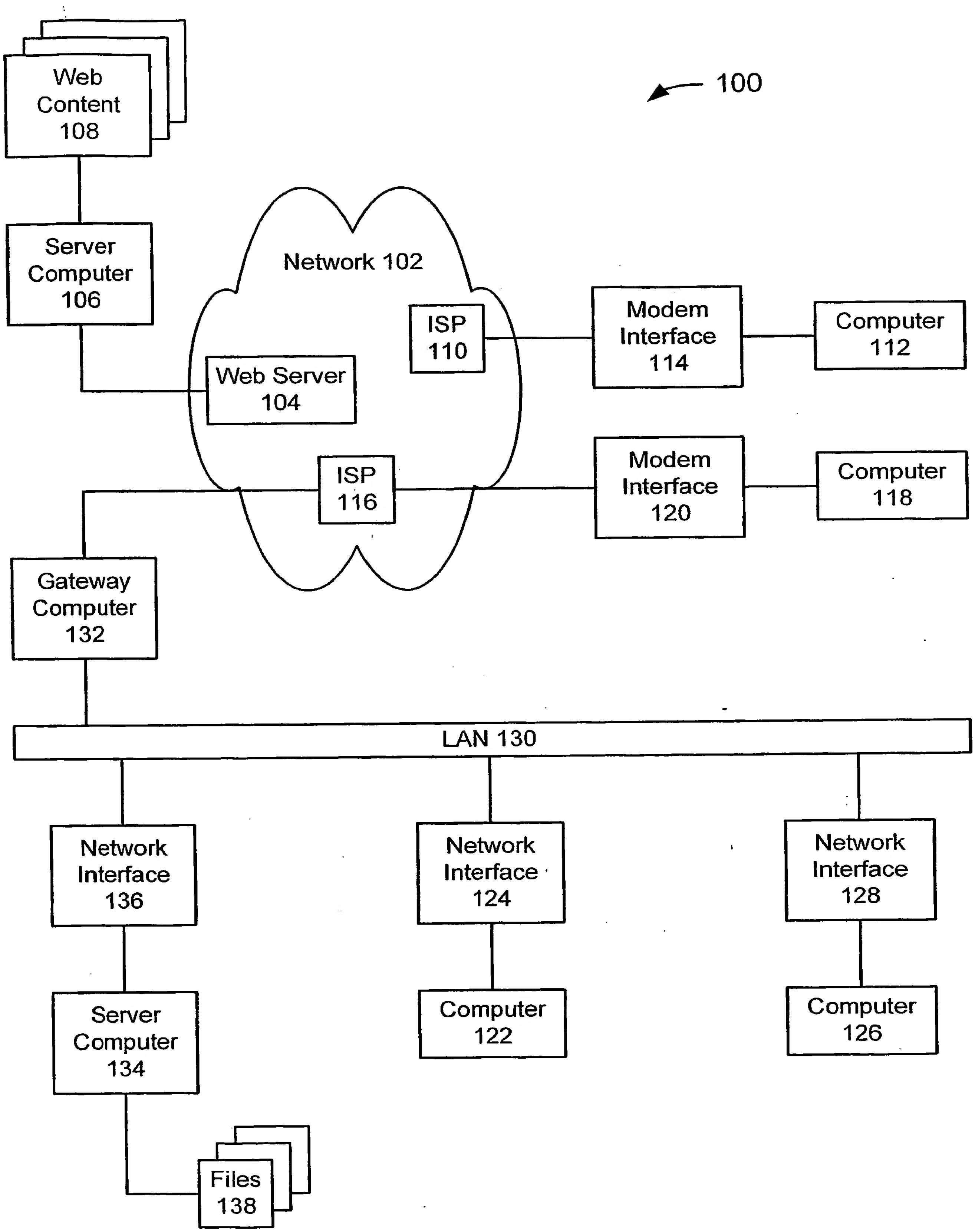
100
Web Content 108
Server Computer 106
Network 102
ISP 110
Web Server 104
ISP 116
Modem Interface 114
Computer 112
Modem Interface 120
Computer 118
Gateway Computer 132
LAN 130
Network Interface 136
Network Interface 124
Network Interface 128
Server Computer 134
Computer 122
Computer 126
Files 138

FIG. 1

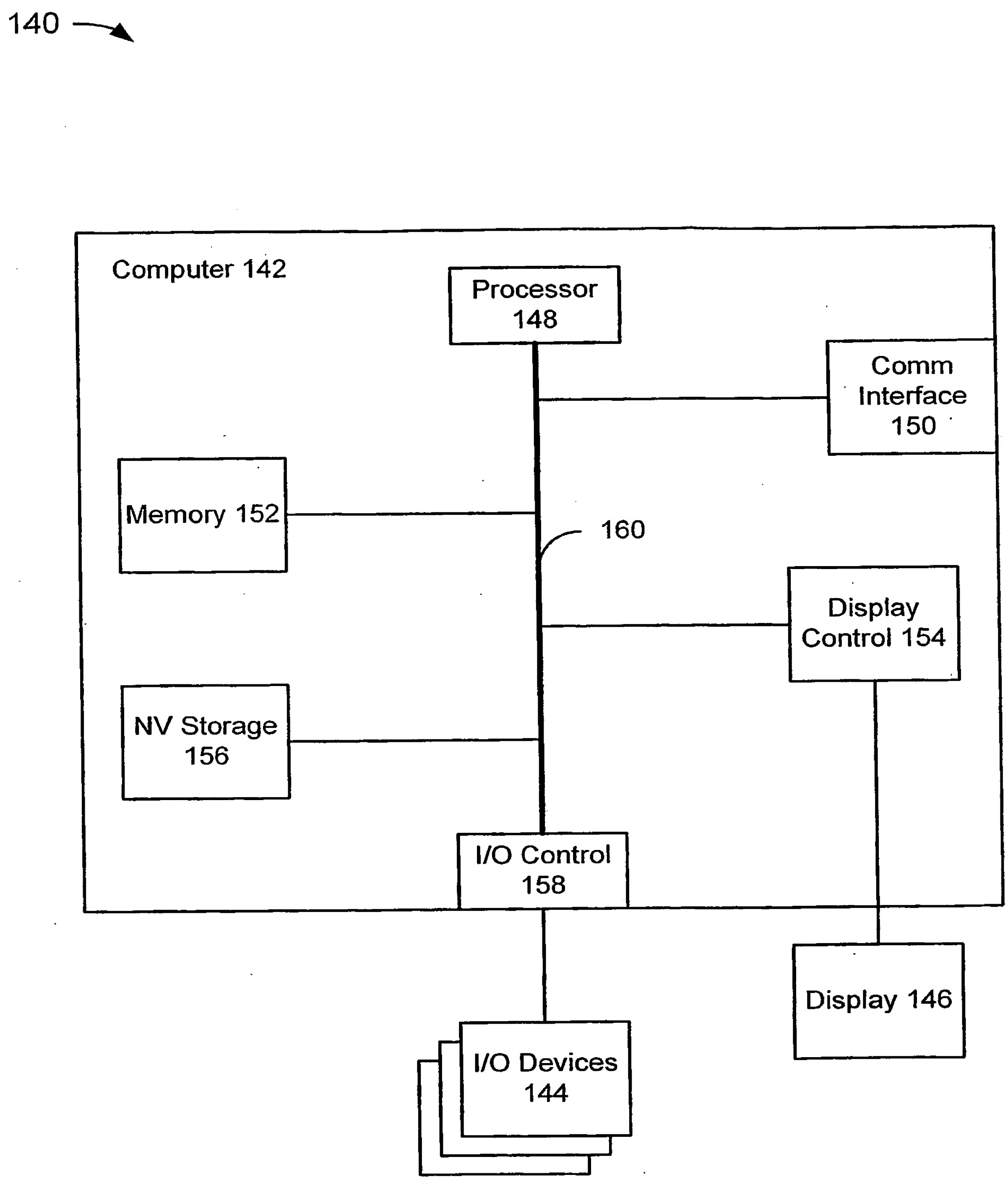
140
Computer 142
Processor 148
Comm Interface 150
Memory 152
160
Display Control 154
NV Storage 156
I/O Control 158
Display 146
I/O Devices 144

FIG. 2

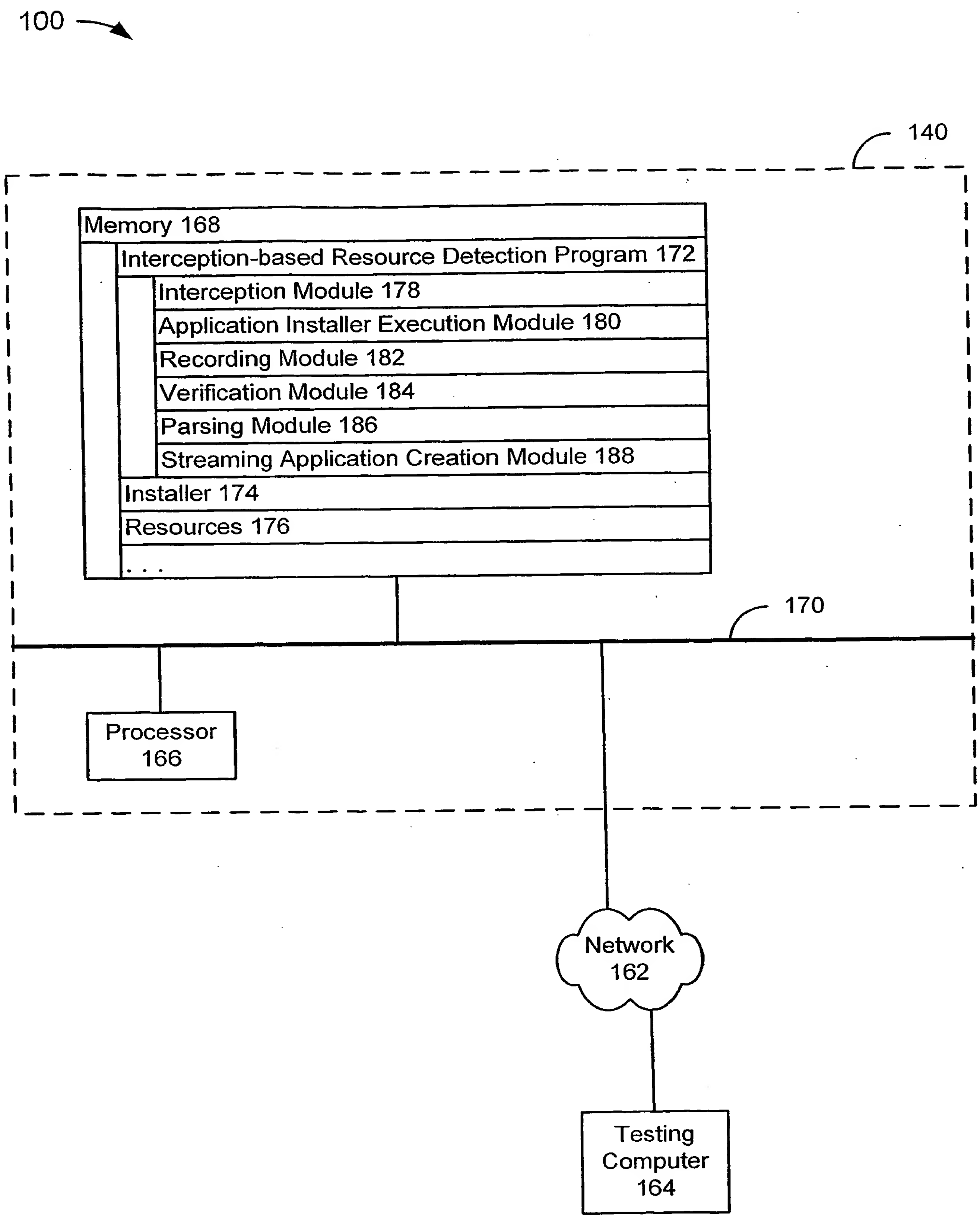
100
140
Memory 168
Interception-based Resource Detection Program 172
Interception Module 178
Application Installer Execution Module 180
Recording Module 182
Verification Module 184
Parsing Module 186
Streaming Application Creation Module 188
Installer 174
Resources 176
. . .
170
Processor 166
Network 162
Testing Computer 164

FIG. 3

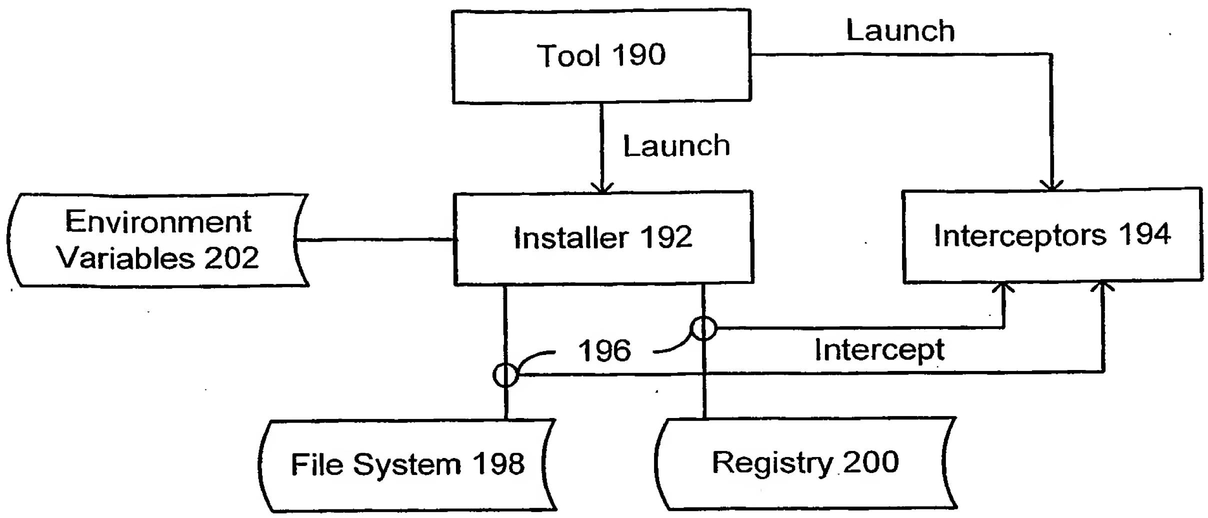

FIG. 4A

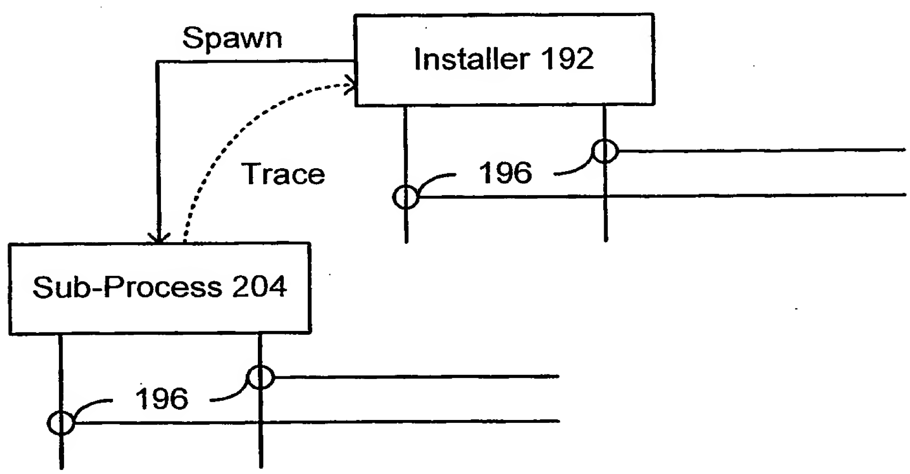

FIG. 4B

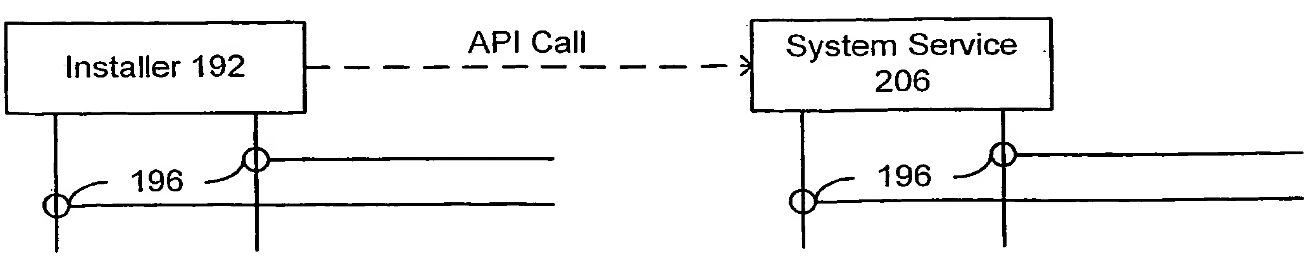

FIG. 4C

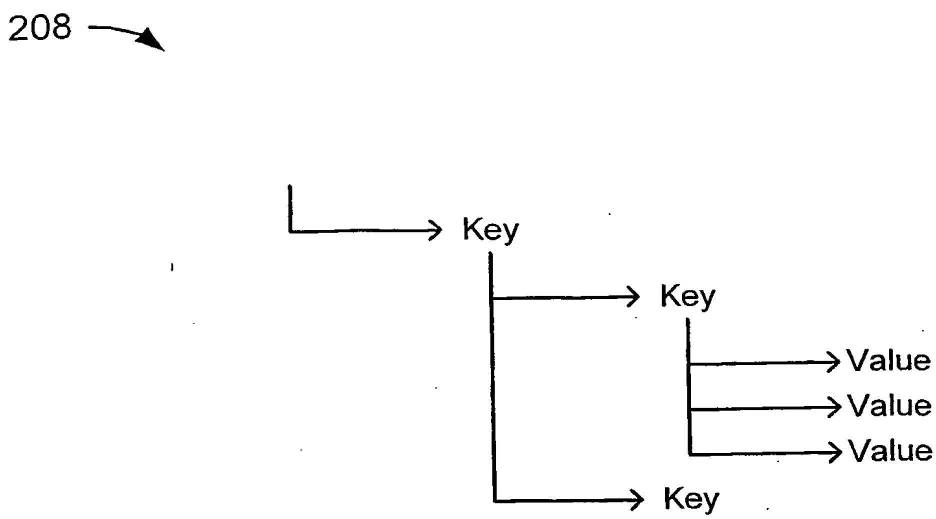
208
Key
Key
Value
Value
Value
Key

FIG. 5A

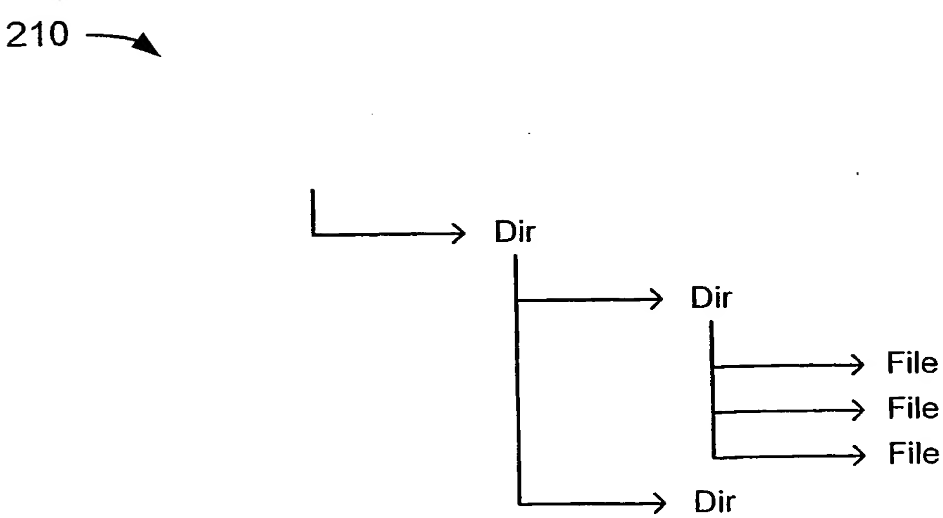
210
Dir
Dir
File
File
File
Dir

FIG. 5B

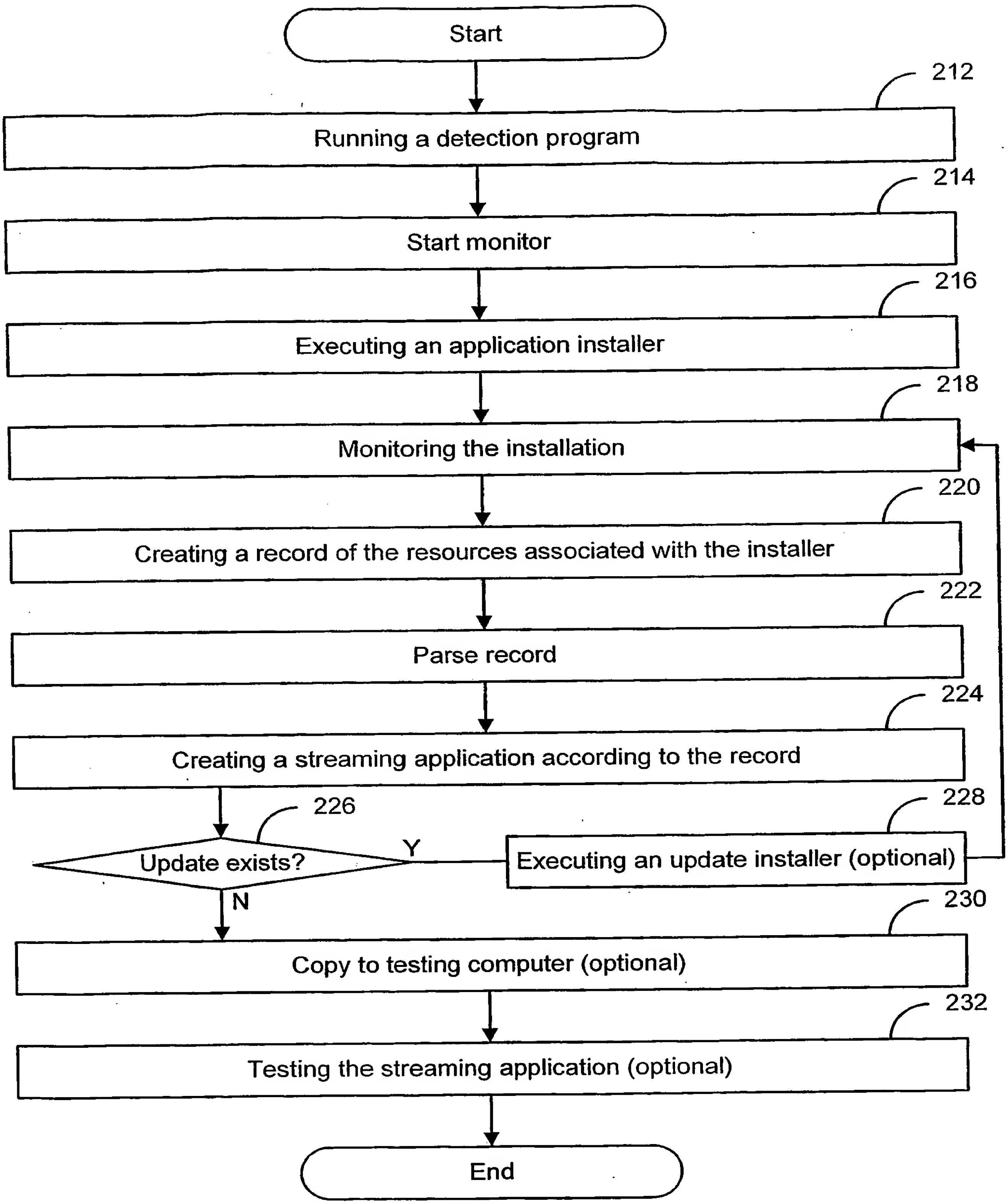
Start
212
Running a detection program
214
Start monitor
216
Executing an application installer
218
Monitoring the installation
220
Creating a record of the resources associated with the installer
222
Parse record
224
Creating a streaming application according to the record
226
Update exists?
Y
N
228
Executing an update installer (optional)
230
Copy to testing computer (optional)
232
Testing the streaming application (optional)
End

FIG. 6

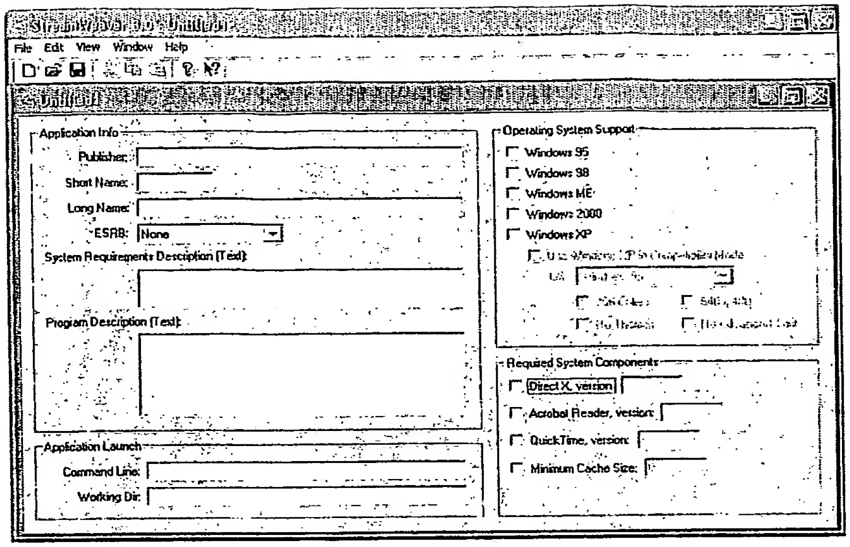
File Edit View Window Help
Application Info
Publisher:
Short Name:
Long Name:
ESRB: None
System Requirements Description (Text)
Program Description (Text)
Application Launch
Command Line:
Working Dir:
Operating System Support
Windows 95
Windows 98
Windows ME
Windows 2000
Windows XP
Required System Components
Direct X, version:
Acrobat Reader, version:
QuickTime, version:
Minimum Cache Size:

FIG. 7A

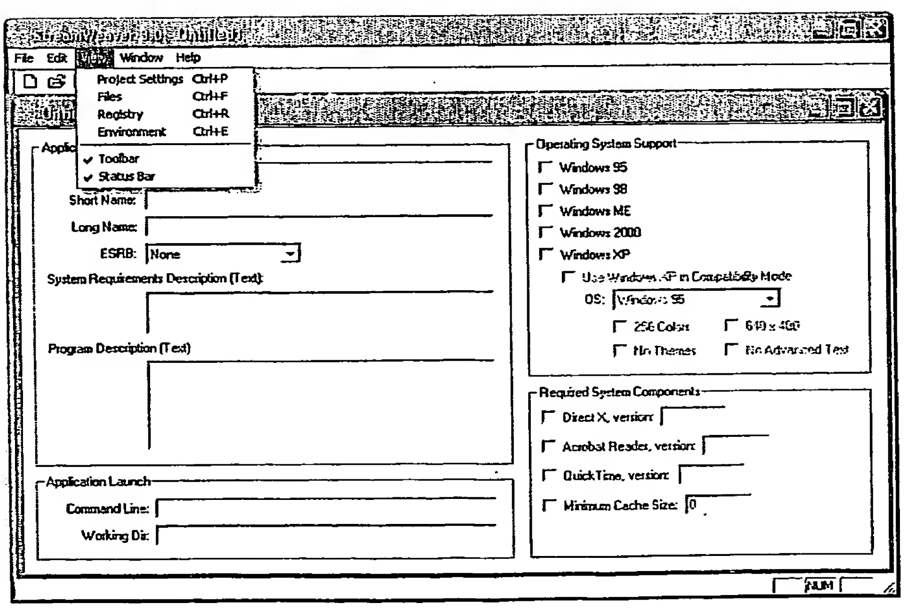
File Edit View Window Help
Project Settings Ctrl+P
Files Ctrl+F
Registry Ctrl+R
Environment Ctrl+E
Toolbar
Status Bar
Short Name:
Long Name:
ESRB: None
System Requirements Description (Text)
Program Description (Text)
Application Launch
Command Line:
Working Dir:
Operating System Support
Windows 95
Windows 98
Windows ME
Windows 2000
Windows XP
Use Windows XP in Compatibility Mode
OS: Windows 95
256 Colors
640 x 480
No Themes
No Advanced Text
Required System Components
Direct X, version:
Acrobat Reader, version:
QuickTime, version:
Minimum Cache Size: 0
NUM

FIG. 7E

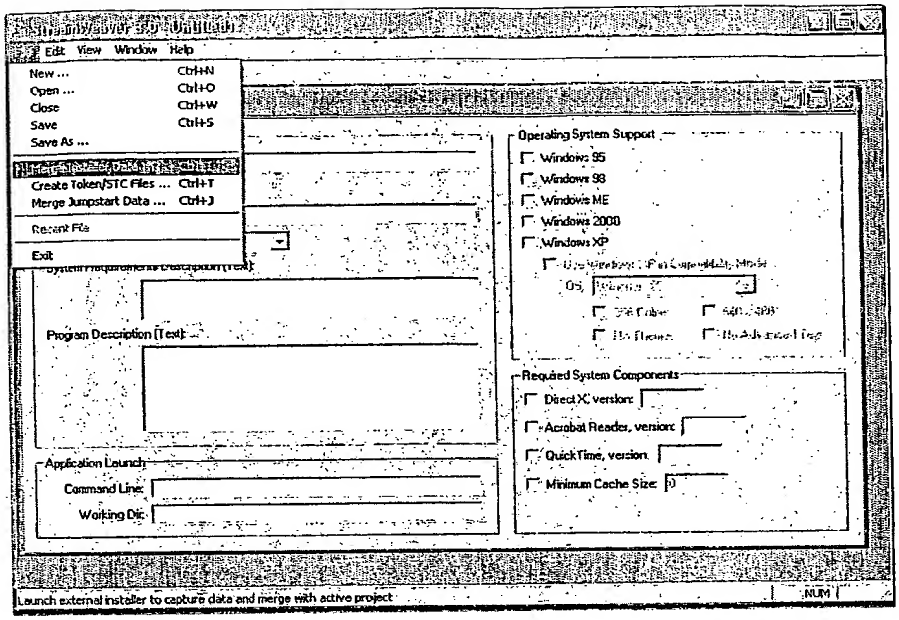

FIG. 7B

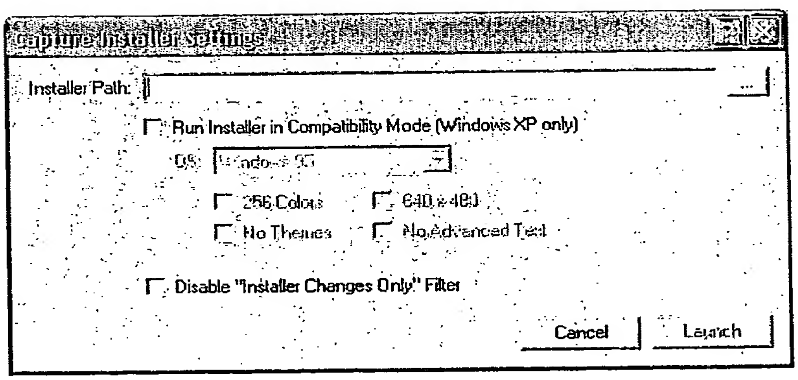

FIG. 7C

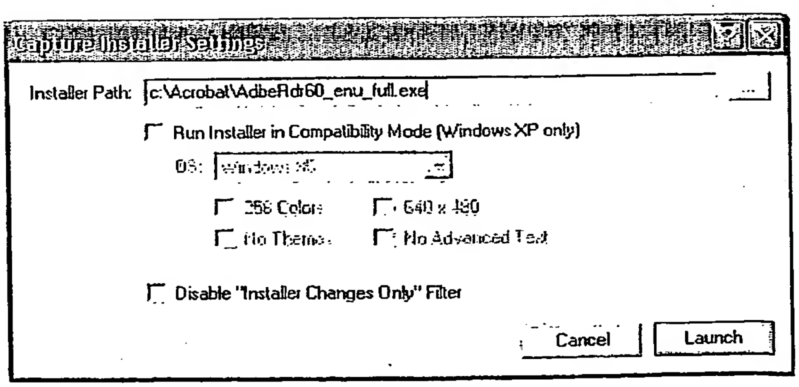

FIG. 7D

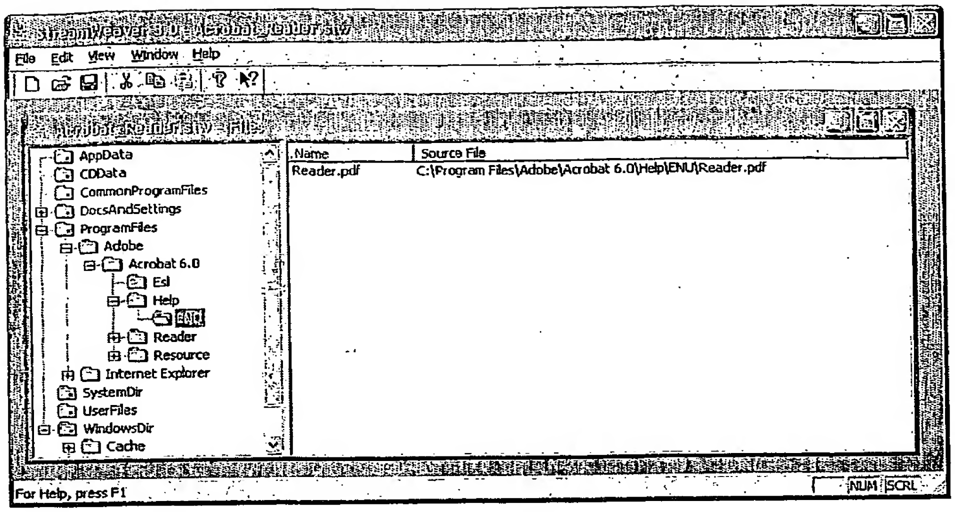

FIG. 7F

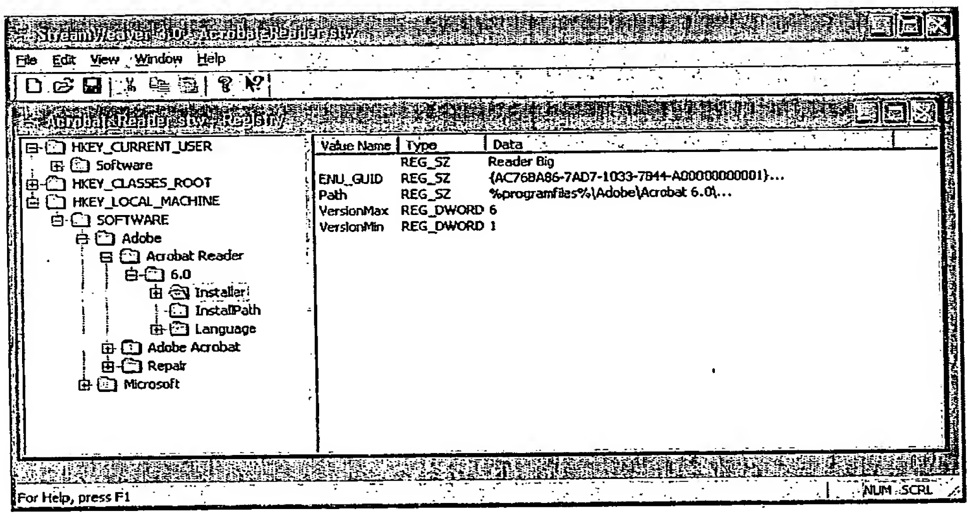

FIG. 7G

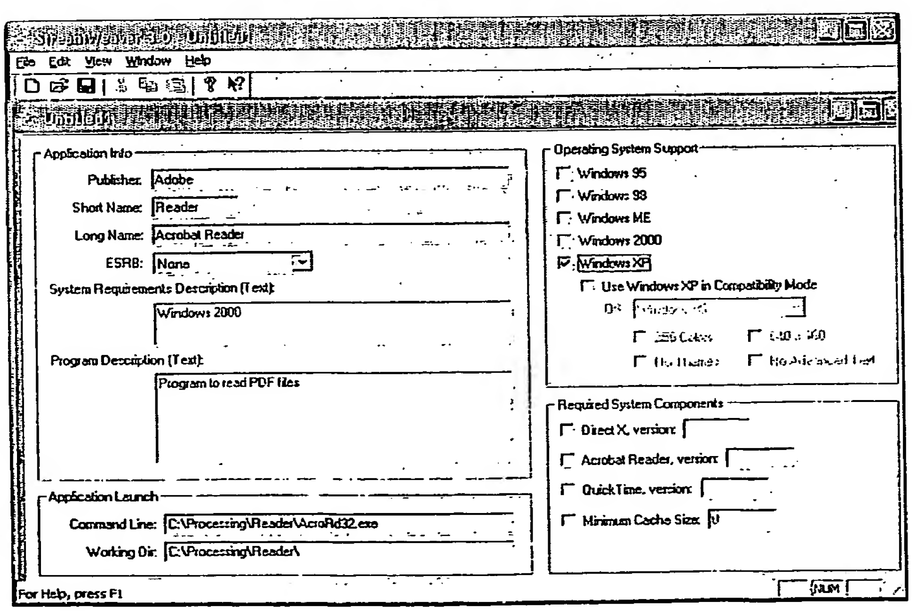

FIG. 7H

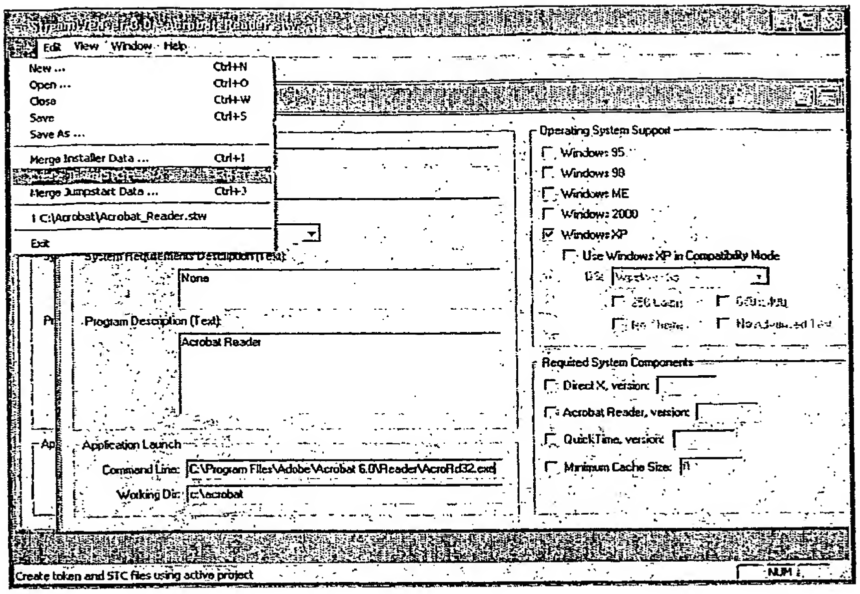

FIG. 7I

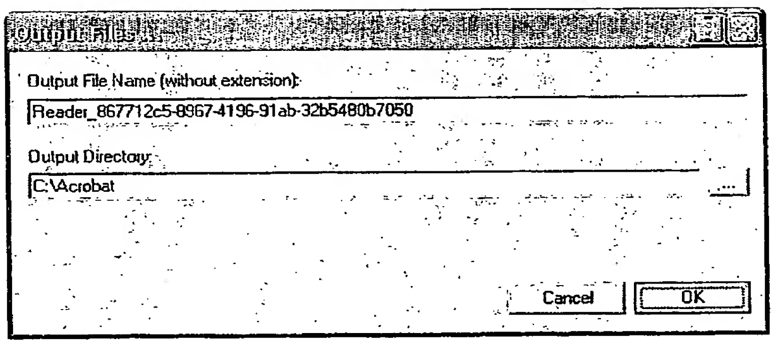

FIG. 7J

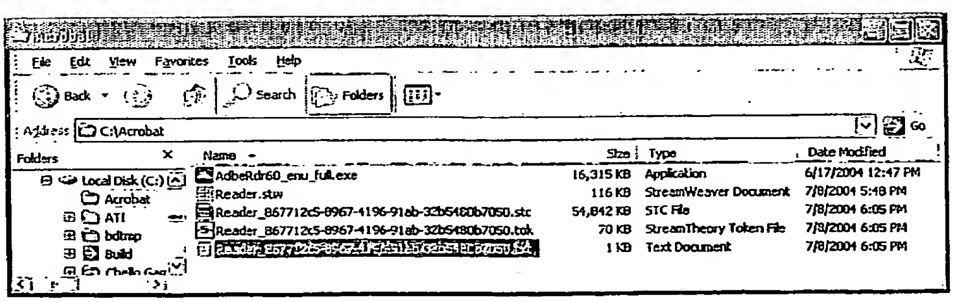

FIG. 7K

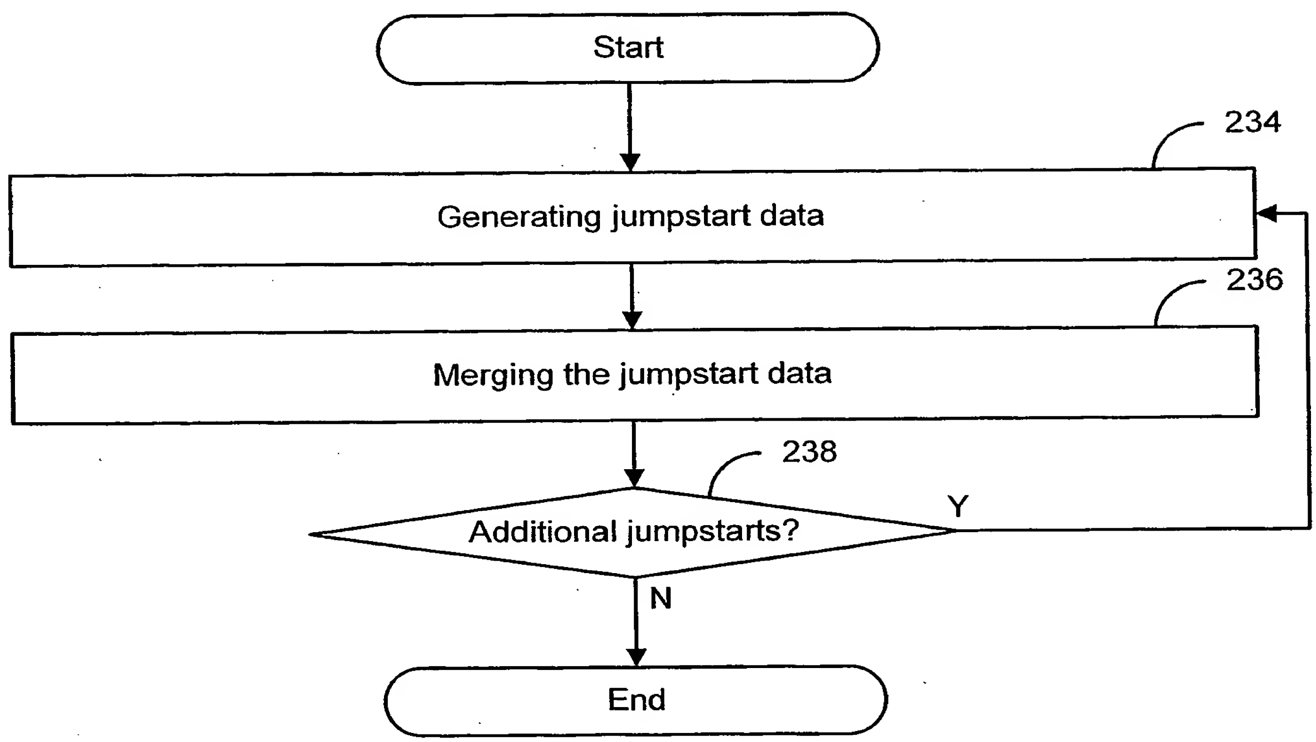
Start
234
Generating jumpstart data
236
Merging the jumpstart data
238
Y
Additional jumpstarts?
N
End

FIG. 8

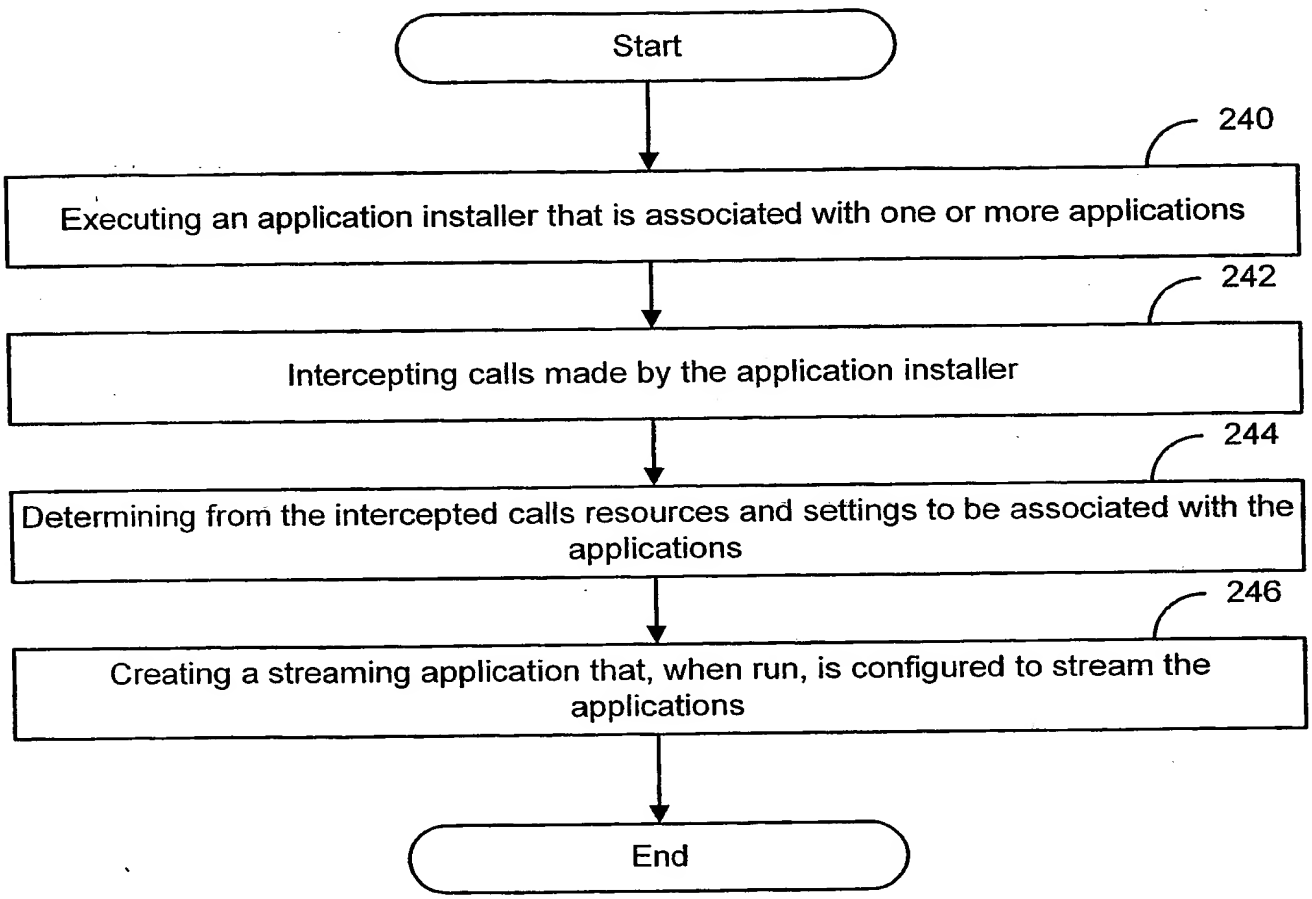
Start
240
Executing an application installer that is associated with one or more applications
242
Intercepting calls made by the application installer
244
Determining from the intercepted calls resources and settings to be associated with the applications
246
Creating a streaming application that, when run, is configured to stream the applications
End

FIG. 9